RasPi

DESIGN
BUILD
CODE

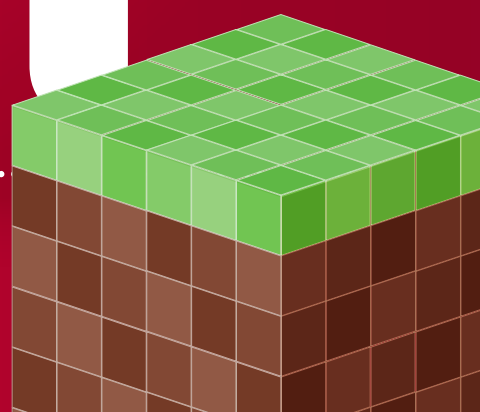
37

Get hands-on with your Raspberry

+ BOOT PI FROM USB

CREATE A PI-POWERED VR SETUP

Plus Draw 3D text in Minecraft



Welcome

"Head-mounted displays are going to be like toasters," said Dr Albert 'Skip' Rizzo, Director of Medical Virtual Reality at the University of Southern California's Institute for Creative Technologies back in 2016. "You might not use it every day but everybody's going to have one." While Oculus Rift isn't quite as ubiquitous as Morphy Richards just yet, there's no denying that VR is certainly set to usher in an entirely new era of home entertainment.

Our main tutorial this issue will help you join this technological revolution using nothing more than the amazingly versatile Raspberry Pi 3, an Xbox 360 controller and some Python code. So swipe on to get all "William Gibson" with your Pi.

Also in this issue you can learn how to draw 3D text Minecraft and boot Pi from a USB drive along with creating a wireless access point.

Get inspired

Discover the RasPi community's best projects

Expert advice

Got a question? Get in touch and we'll give you a hand

Easy-to-follow guides

Learn to make and code gadgets with Raspberry Pi

Dave
Editor

From the makers of
LinuxUser & Developer

Join the conversation at...

@linuxusermag | Linux User & Developer | RasPi@futurenet.com

Contents

Create a Pi-powered virtual reality setup

Combine the Raspberry Pi, Python-VRZero and 3D graphics module Pi3D to edit or make virtual reality environments

Virtual Reality is huge now and has come a long way since the concepts and CGI visuals of Stephen King's Lawnmower Man. It is one of the fastest growing areas of technology and you can now design models, explore new places and play games all within a virtual environment.

A professional VR hardware package is expensive and will set you back several hundred pounds. However, it's possible to emulate the VR setup up using a Raspberry Pi, Python-VRZero and a 3D graphics module, Pi3D. Now, this is purely for fun and learning, so don't expect huge gaming PC frame rates, although some of the demos do peak at around 25-30 FPS on a Raspberry Pi 3. This tutorial shows you how you create a VR setup using the Raspberry Pi 3, a Xbox 360 controller and some Python code. Our first steps will walk you through how to install the required software and modules. We'll then cover configuring the hardware and drivers to enable you to control movement within the VR environment. The final steps look at the program code structure, where you can develop your own versions of the VR demo or design and build your own virtual worlds.

THE PROJECT ESSENTIALS

Raspberry Pi 3

8GB SD card

Xbox 360 controller

Oculus Rift Developer Kit (optional)

01 Python-VRZero

Using Python-VRZero is a frictionless way to get started creating your own virtual reality worlds in Python on a Raspberry Pi and combine an Oculus Rift. The program adds a number of features on top of Pi3D and solves the headaches of configuring a Pi 3 for VR development in Python. It includes default input event handling for keyboard, mouse, controller and the Rift for moving and altering the view of the player. It also supports Xbox 360 controller configuration and uses OpenHMD to read the rotational sensor reading from the Rift.

02 Fresh SD card install

Before you get started, it is recommended that you use a new SD card with a fresh installed image of the Raspberry Pi's official operating system, Raspbian. You can download the operating system directly from the Raspberry Pi website at **https://www.raspberrypi.org/downloads**. Install using your normal preferred method. This project setup was tested using the Pixel OS image.

03 Update the Pi

Boot up your Raspberry Pi. At this stage you do not need to connect the Xbox 360 controller or Oculus Rift. When loaded, open the LXTerminal and update and upgrade the OS typing the two lines below. This may take some time.

```
sudo apt-get update
sudo apt-get upgrade
```

04 Install the Xbox 360 controller drivers

Now install the package dependencies for the Xbox 360 controller. You can keep the library up to date by adding the code `--upgrade` to the end of the first line. Then install Pi3D software which will render the images. Type the two lines below as shown and press Enter.

```
sudo apt-get install -y libhidapi-libusb0 xboxdrv
sudo pip3 install pi3d==2.14
```

05 Install the VR software – part 1

The Python-VRzero is available from the GitHub website and is easily downloaded using the `git clone` command, line one. Type this into your LXTerminal. Then move to the **python-vrzero** folder (line two) and install the program (line three). This deploys the software to interact between the hardware, 3D software and VR environment.

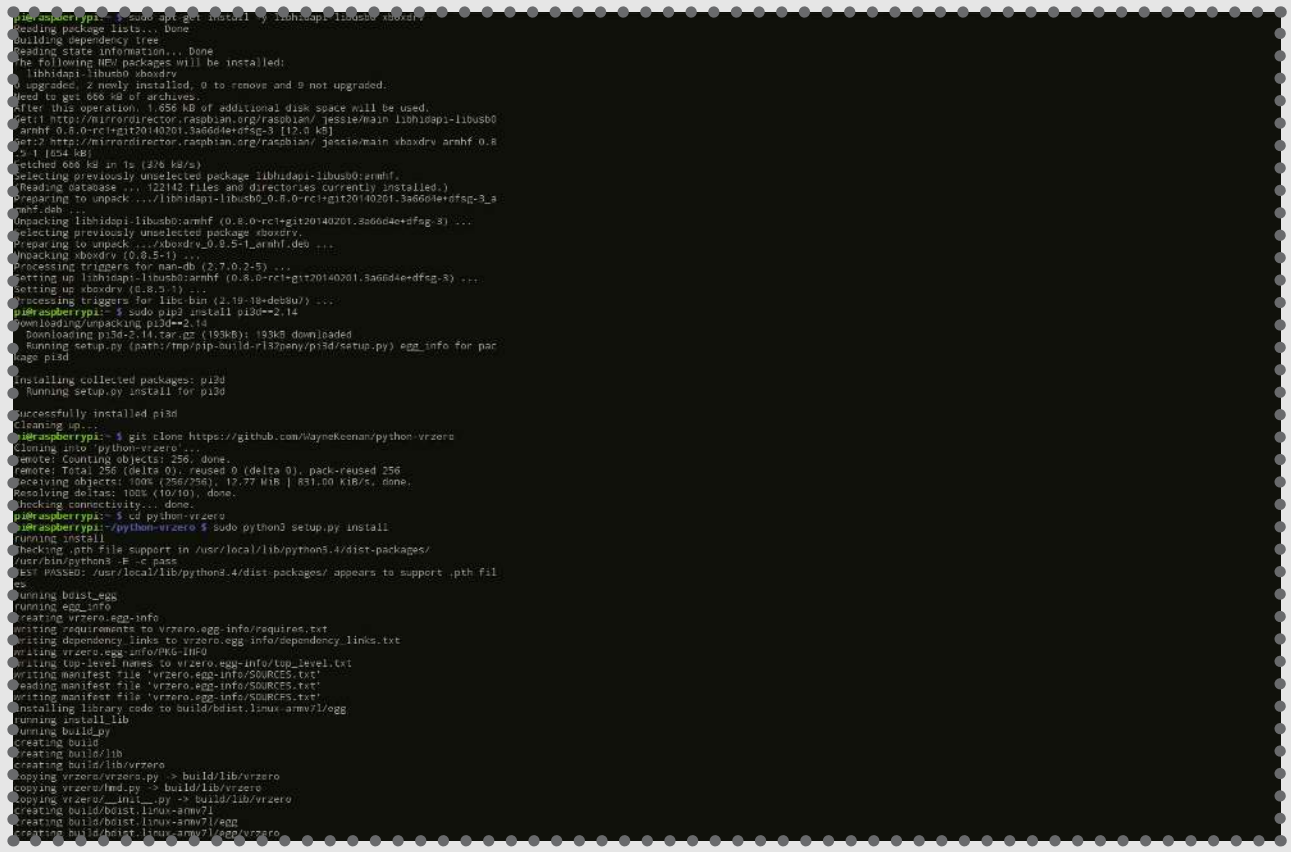
```
sudo git clone https://github.com/WayneKeenan/python-vrzero
cd python-vrzero
sudo python3 setup.py install
```

06 Install the VR software – part 2

Once the installation completes, select and install the OpenHMD (line one) which enables the data from the Oculus Rift sensors to be read and worked with. Type line one into the LXTerminal and press Enter, then line two to install the required module. Enter line three and press Enter to link together all the required libraries:

Try some other demos

You may be interested in trying out some other demonstrations which are availble at **https://github.com/pi3d/pi3d_demos.** Perhaps try riding a Roller Coaster or driving a tank! This resource also provides a guide how to create your own models using the Pi3D python library and code.




```
sudo dpkg -i install/openhmd_0.0.1-1_armhf.deb
sudo apt-get install -f
sudo ldconfig
```

07 Copy over the configuration files – part 1

To interact with the Oculus Rift's rotation sensor and the Xbox 360 controller, you'll need to copy over the configuration files. This enables you to orientate and look around the environments. As you turn your head to the left the VR environment will adjust as if you are looking to the left. Type both of the lines below into the LXTerminal and press Enter after each line:

```
sudo cp config/83-hmd.rules /etc/udev/rules.d/
sudo cp config/xboxdrv.init /etc/init.d/xboxdrv
```

08 Copy over the configuration files part 2

The Xbox 360 controller setup requires an additional command line to copy the default configuration file to the folder which contains the Xbox Drivers. This file contains all the

```
Adding numpy 1.8.2 to easy-install.pth file

Using /usr/lib/python3/dist-packages
Searching for pi3d==2.14
Best match: pi3d 2.14
Adding pi3d 2.14 to easy-install.pth file

Using /usr/local/lib/python3.4/dist-packages
Finished processing dependencies for vrzero==0.0.1
pi@raspberrypi:~/python-vrzero $ sudo dpkg -i install/openhmd_0.0.1-1_armhf.deb
Selecting previously unselected package openhmd.
(Reading database ... 122196 files and directories currently installed.)
Preparing to unpack .../openhmd_0.0.1-1_armhf.deb ...
Unpacking openhmd (0.0.1-1) ...
Setting up openhmd (0.0.1-1) ...
pi@raspberrypi:~/python-vrzero $ sudo apt-get install -f
Reading package lists... Done
Building dependency tree
Reading state information... Done
0 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
pi@raspberrypi:~/python-vrzero $ sudo ldconfig
pi@raspberrypi:~/python-vrzero $ sudo cp config/83-hmd.rules /etc/udev/rules.d/
pi@raspberrypi:~/python-vrzero $ sudo cp config/xboxdrv.init /etc/init.d/xboxdrv
```

mapping for the buttons, paddles and joysticks. Type in the line as shown below and press Enter:

```
sudo cp config/xboxdrv.defaults /etc/default/xboxdrv
```

09 Rift Development Kit 1

If you do not have an Oculus Rift kit you can still use a HDMI monitor to display the output. Move to Step 11. If you own or have access to the Oculus Rift Development kit, you will need to copy over the configuration file into the config.txt file. This file contains the configuration settings for the operating system which are loaded when you Raspberry Pi boots up. Type the line below into the LXTerminal and press Enter.

```
sudo cp config/config_DK1.txt /boot/config.txt
```

10 Rift Development Kit 2

If you have access to the Oculus Rift development kit version 2, then you will again be required to copy over a configuration file. Except this time select the config_DK2.txt file and copy the contents to the boot/config file. In the LXTerminal type the line below and press enter. Ensure

"If you do not have an Oculus Rift kit you can still use a HDMI monitor to display the output"

```
Processing dependencies for vrzero==0.0.1
Searching for numpy==1.8.2
Best match: numpy 1.8.2
Adding numpy 1.8.2 to easy-install.pth file

Using /usr/lib/python3/dist-packages
Searching for pi3d==2.14
Best match: pi3d 2.14
Adding pi3d 2.14 to easy-install.pth file

Using /usr/local/lib/python3.4/dist-packages
Finished processing dependencies for vrzero==0.0.1
pi@raspberrypi:~/python-vrzero $ sudo dpkg -i install/openhmd_0.0.1-1_armhf.deb
Selecting previously unselected package openhmd.
(Reading database ... 122196 files and directories currently installed.)
Preparing to unpack .../openhmd_0.0.1-1_armhf.deb ...
Unpacking openhmd (0.0.1-1) ...
Setting up openhmd (0.0.1-1) ...
pi@raspberrypi:~/python-vrzero $ sudo apt-get install -f
Reading package lists... Done
Building dependency tree
Reading state information... Done
0 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
pi@raspberrypi:~/python-vrzero $ sudo ldconfig
pi@raspberrypi:~/python-vrzero $ sudo cp config/83-hmd.rules /etc/udev/rules.d/
pi@raspberrypi:~/python-vrzero $ sudo cp config/xboxdrv.init /etc/init.d/xboxdrv
pi@raspberrypi:~/python-vrzero $ sudo cp config/xboxdrv.defaults /etc/default/xboxdrv
pi@raspberrypi:~/python-vrzero $ sudo cp config/config_DK1.txt /boot/config.txt
pi@raspberrypi:~/python-vrzero $ sudo udevadm control --reload-rule
pi@raspberrypi:~/python-vrzero $ sudo systemctl disable hciuart
```

that you select the correct configuration for the kit version which you have.

```
sudo cp config/config_DK2.txt /boot/config.txt
```

11 Complete the set up

Finally run two commands. The first command (line one) enables the root-less USB udev config setup which was set up earlier in Step 7. The second command (line two) disables BluetoothLE. This is required as it stops the OpenGL ES, from hanging. Ensure that each line is typed in as printed, and press Enter after each line to enable the command to run:

```
sudo udevadm control --reload-rules
sudo systemctl disable hciuart
```

12 Restart and plug in

This completes the installation and project setup. From the LXTerminal, shutdown the Raspberry Pi (line one). Attach the Xbox 360 controller and if you have one, the Oculus Rift. Turn the Rift on first before starting the Pi to

"Ensure that each line is typed in as printed, and press Enter after each line to enable the command to run"

ensure that it registers the hardware when the Pi boots up:

```
sudo shutdown
```

13 Hardware controls and setup

Python-VRzero sets up sensible defaults for handling input events from attached devices. The keyboard controls movement and the default mappings are: WSAD, SPACE for jump and ENTER for action. The mouse controls looking (and direction of travel when moving). Mouse button 1 is action and mouse button 2 is jump. An Xbox 360 controller controls movement and view using the left and right stick respectively. The 'A' button is 'action' and the 'B' button is jump. The OpenHMD library is used to read the HMD sensor data. VR Zero automatically rotates the Pi3D Steroscopic camera in response to HMD readings.

14 Running a demo

Now for the fun part, which is to run a demo program and immerse yourself in a VR world. There are several to choose from, and each one demonstrates the features of the Python–VRZero program. The demos need to be run using Python 3 and executed as a script from the demos folder. (If you are using a Oculus Rift you will need to navigate to the folder via the display inside the Rift headset.) Open the LX Terminal, and move to the python-vrzero/demos folder, line one. To list the available demos type ls, this will list the file names of all the demos. To run a demo type ./ followed by the name of the demo, for example to run the abbey demo type, ./abbey.py (line 2). You will be presented with a VR render of Buckfast Abbey, to end the environment press Escape on the keyboard.

```
cd python-vrzero/demos
```

```
./abbey.py
```

15 Editing the textures

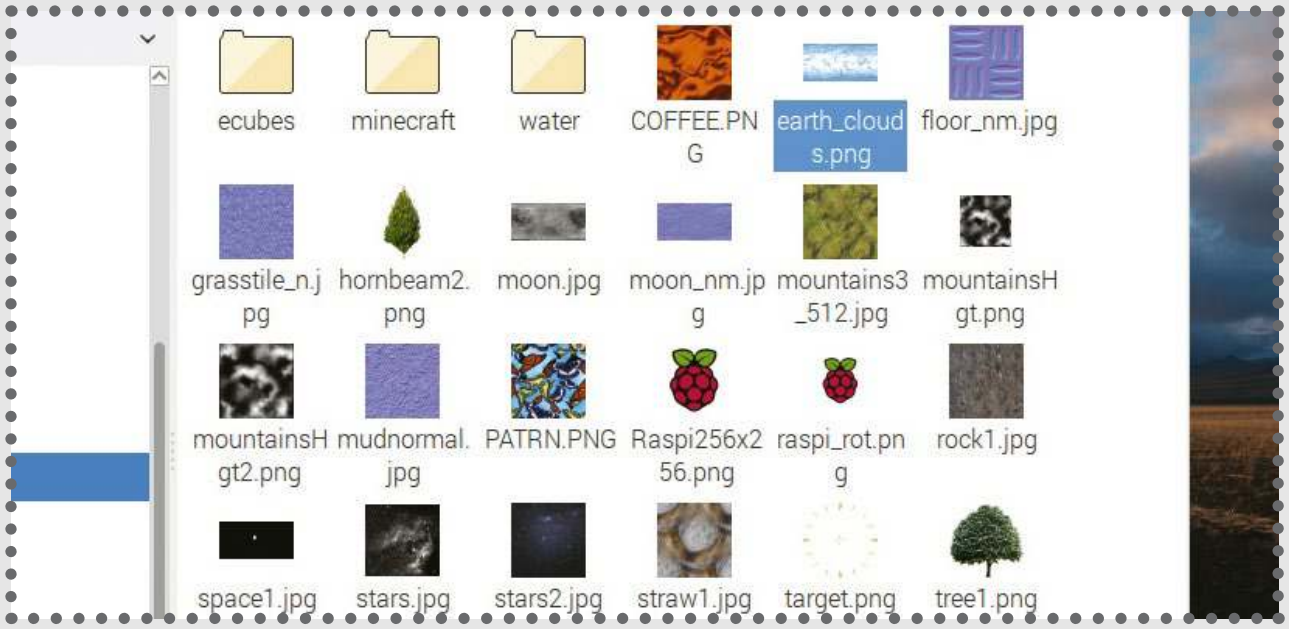
If you have used Pi3D before you can access the program template to set up your own models. If not, then change the textures in the Shape demo program. First open the LXTerminal and type sudo idle 3 to load Python 3. Select File and open, navigate to the following folder / home/pi/python-vrzero/demos and select the shapes. py program. Locate line 14 which begins patimg = pi3d (pictured, top of the page). This is the first line of code which tells the program which textures to load for each shape. There are several after which can also be edited.

16 Change a texture

Using the folder explorer, click the file icon to navigate to the textures in the texture folder /home/pi/python-vrzero/demos/textures. You will see the files that are used for the shapes demo. Replace the image file with one of your own and then on line 14 of the program change the file name to match your selected image file choice. If you don't want to use your own texture file then you can change the file name to one of the other image files listed in the folder. Press F5 to save and run the demo. You will notice that the shape textures have changed.

17 Alter the message

Return to your Python editor and locate line 71. This holds the message which is displayed in the shapes VR demo. Change the text to a sentence of your choice.Save and run the program. Congratulations you have now begun to modify your own VR demos. Experiment with the program files for each of the demos editing the textures.



For example, how about creating a church made out of chocolate? If you want to try other demos, find additional details at **https://github.com/pi3d/pi3d_demos**.

```
# also inside surfaces need to be defined otherwise normals are wrong
mylathe = pi3d.Lathe(path=((0.0, 1.0), (0.6, 1.2), (0.8, 1.4), (1.09, 1.7),
                           (1.1, 1.7), (0.9, 1.4), (0.7, 1.2), (0.08, 1), (0.08, 0.21),
                           (0.1, 0.2), (1.0, 0.05), (1.0, 0.0), (0.001, 0.0), (0.0, 0.0)),
                     sides=24, name="Cup", x=0, y=-1, z=10, sx=0.8, sy=0.8, sz=0.8)
mylathe.set_draw_details(matsh, [shapebump, shapeshine], 0.0, 1.0)
mylathe.set_alpha(0.5)
mytorus = pi3d.Torus(radius=1, thickness=0.3, ringrots=12, sides=24, name="Torus",
                     x=2, y=-1, z=10)
myhemisphere = pi3d.Sphere(radius=1, sides=24, slices=24, hemi=0.5, name="hsphere",
                           x=4, y=-1, z=10)
myPlane = pi3d.Plane(w=4, h=4, name="plane", z=12)
# Load ttf font and set the font colour to 'raspberry'
arialFont = pi3d.Font("fonts/FreeMonoBoldOblique.ttf", (221,0,170,255))
mystring = pi3d.String(font=arialFont, string="Now the Raspberry Pi really does rock", z=4)
mystring.set_shader(flatsh)

angl = 0.0


engine.avatar_position = [0.0, -5.0, 0.0]


def draw():
    mysphere.draw(shader, [patimg])
    mysphere.rotateIncY(2.5)
    mysphere.rotateIncX(0.6101)

    myhemisphere.draw(shader, [coffimg])
    myhemisphere.rotateIncY( .5 )

    myhelix.draw(shader, [patimg])
    myhelix.rotateIncY(3)
    myhelix.rotateIncZ(1.1)

    mytube.draw(shinesh, [coffimg, shapebump, shapeshine], 4.0, 0.1)
    mytube.rotateIncY(3)
    mytube.rotateIncZ(2)

    # Extrude has different textures for each Buffer so has to use
    # set_draw_details() above, rather than having arguments passed to draw()
    myextrude.draw()
    myextrude.rotateIncY(-2)
```

Tough Pi-ano

Bryan McEvoy's tough-as-nails piano takes an all-new approach to music exploration

Where did your inspiration for the Tough Pi-ano come from?

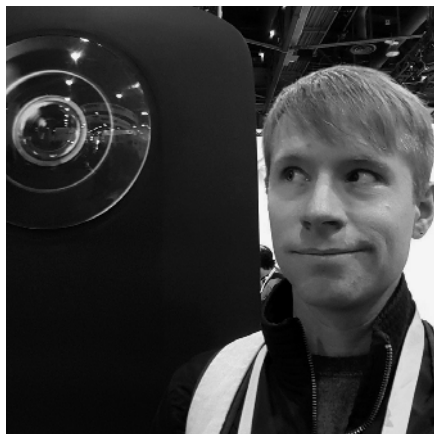
I started the Tough Pi-ano after talking with my aunt. My cousin loves music therapy, particularly playing the piano, but he has Down's syndrome and his excitable therapy sessions can destroy instruments meant purely for gentle fingers. While I thought of a solution, he kept smashing every garage sale piano they were able to pick up and buy. I knew that each of those pianos was really just a series of inputs and a rather basic sound generator.

Has it been easy getting the project from concept to working device?

One year ago I started working on a version which used hardwood keys with the same proportions as a standard piano, but they were difficult to make in my local hackspace – so I had to rethink this idea.

A couple months after the piano had been left aside, I decided it would be better to use commercial parts for the keys even if they didn't look like a piano. After all, I was building this for kids who wanted to smash buttons, not concert pianists. I ordered a ton of arcade buttons in black and white, which obviously give that natural piano look.

Replaceable octaves didn't make it to the final version, but using one computer for each octave stayed. Even if someone managed to destroy one of the computers, three-quarters of the piano would keep running. It would also be possible to have spares on hand until a proper repair could be made. All the arcade switches were connected with slide-on terminals so they could be replaced if someone wore it out. Redundancy was probably the strongest quality of the Pi-ano, but you wouldn't see it unless you looked under the hood. Designing a musical instrument with the explicit intention of absorbing abuse was a different kind of

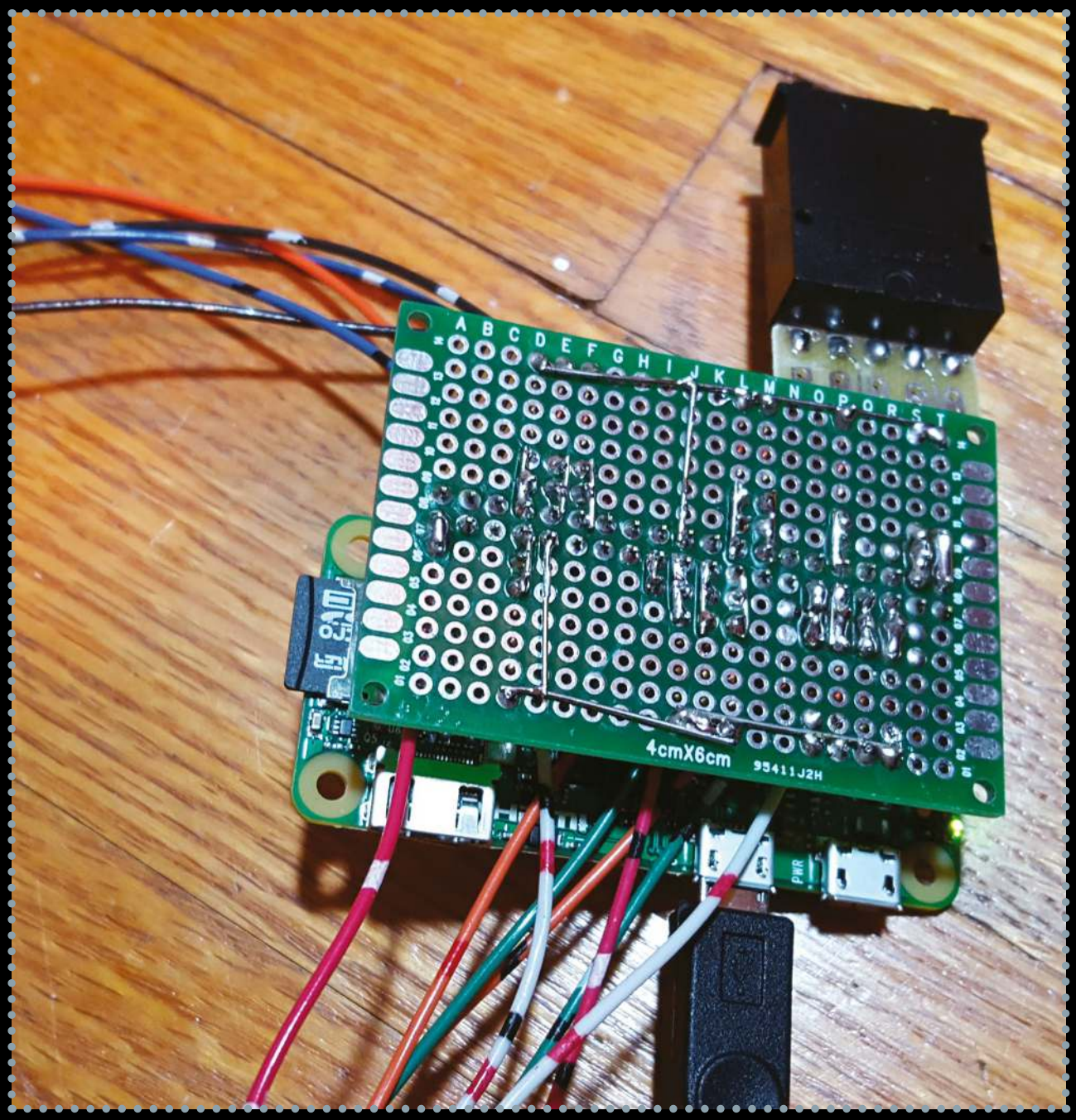


Bryan McEvoy is an automation engineer by trade, which has given him the chance to design and build some pretty incredible things.

brainstorming for me. Many of my previous projects were only meant to be strong enough to withstand ordinary wear and tear. The Tough Pi-ano had to take punishment all day long without hurting anyone and have lots of parts that were easy to service. That's why there's a plastic sheet over the keys: no slivers. It'll be wall mounted so it can't fall on anyone. There is no exposed metal if knuckles start scraping. It was a whole different way to approach the user interface.

How accurately does the Tough Pi-ano reproduce sound?

The sound files for this project were legitimate piano samples and the keys corresponded to the correct notes, so anyone who played the piano could sit down and perform a song, but they would have to reach more since the keys were



THE PROJECT ESSENTIALS

- Raspberry Pi units
- Charging and OTG cable
- Dual sound cards
- Thumbwheel switch
- Headphone cable
- Powered speakers
- Headphone splitter
- Power adaptor
- Hardwood casing
- Buttons

spaced out so much further than a standard piano. Of course, certain limitations are noticeable, but nothing major.

We've seen some mention of using the piano tool within Pygame on your Tough Pi-ano. Did this prove to be a big help?

Pygame was a huge help with this project. The community surrounding Pygame was wonderful, I didn't have to ask them a single question, I was able to find everything I needed in their guides and forums. That kind of support is the same reason I program with the Raspberry Pi. If someone wanted to copy my project, they could do it inexpensively and if they had trouble, there's help available. In fact, someone has already started to build their own Tough Pi-ano for a school and together we discovered that my code won't work on a Pi Zero W.

How much of a learning curve has this project been for you in terms of developing with the Pi and Python?

I knew almost nothing about Python when I started this project and I learned a lot. It was honestly a very good project for a beginner, since you build the same switch circuit 12 times for each octave and then build a total of four octaves. One of my rookie mistakes was building a circuit board to hold pull-up resistors for each of the inputs. Later, I learned this could have been done in software. I won't repeat that mistake in the future.

How happy are you with the finished product? What would you say is its best feature?

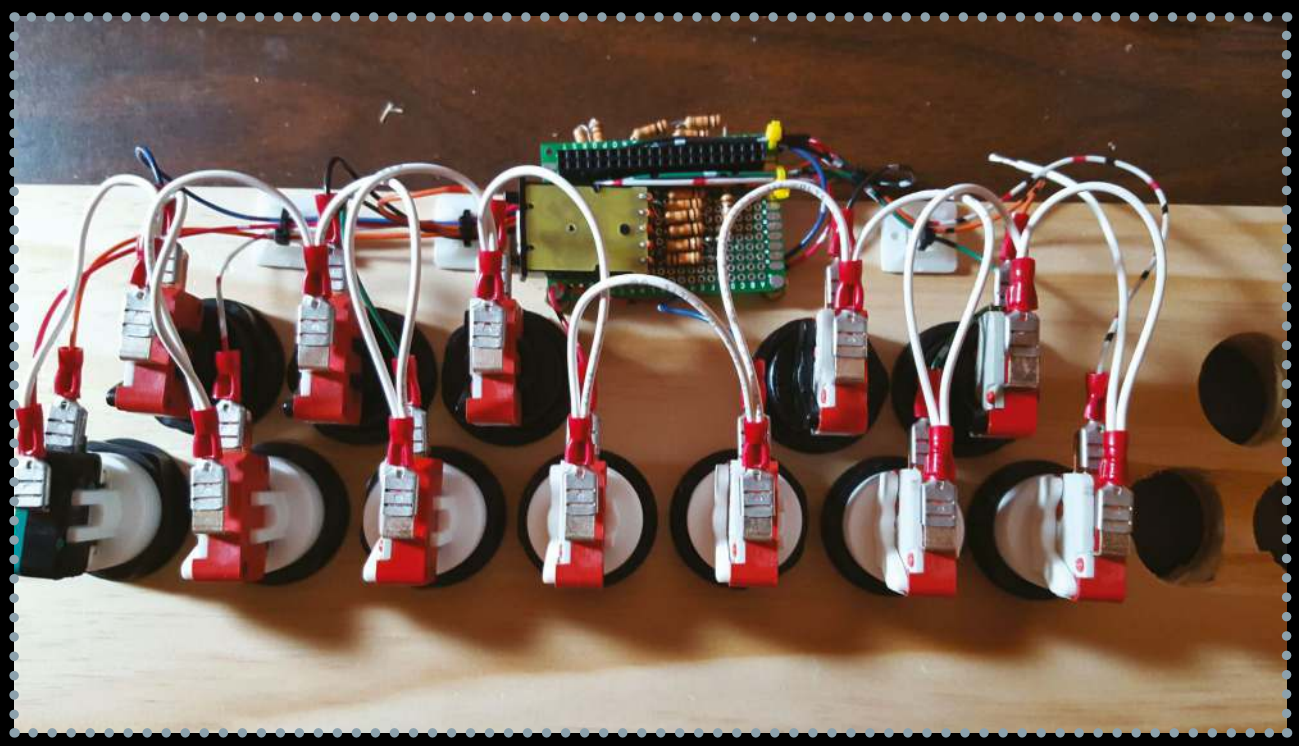
The Tough Pi-ano's best quality, in my opinion, was the simplicity. You put power on it and then you have 48 buttons that make sounds. Nothing extraneous, no settings that

might interrupt a jam session, no knobs to break off, no hinges to pinch fingers. One change I would welcome would be a solid case made of heavy-duty plastic.

My aunt and uncle are opening a facility on their property where other kids with Down's, or on the autism spectrum, can come with their families to spend time in a safe environment and be themselves. I'm pleased that they will have a piano they don't have to be gentle with.

Looking past the Tough Pi-ano, will you be using the Pi to build any other projects in the future?

I think the Raspberry Pi line computers are great pieces of hardware. In the past, I've built a MAME box and a wearable computer with a head-mounted display; it runs on just a USB battery pack. Right now though, I'm using them to build a laser tag game with Wi-Fi connectivity. I like to detail all my projects, including how I build them, on my personal blog (www.24hourengineer.com), as I find it helps me to keep motivated and keep on developing.



Make a Raspberry Pi VPN Access Point

Use your Pi as a Wireless AP that's connected to your VPN 24/7

Anyone who has been abroad will know it's a constant pain to have to deal with content that has been switched to the local language, not to mention certain states that censor some internet content.

Fortunately there's a way to evade these restrictions using the Raspberry Pi. In a few simple steps you can configure your Pi to generate its own wireless AP (Access Point) and keep it permanently connected to a VPN (Virtual Private Network) service of your choosing.

All you need to do when travelling is bring your Pi and connect it to a working router and you'll have your own private wireless network and connection.

VPNs were originally designed to allow office workers to connect to their corporate intranet while away, over an encrypted connection. These days they're more commonly used both to protect your connection and make it seem as if your computer is located in another country.

In order to proceed with this project, you will require an active VPN subscription and the client configuration (.conf) file to automatically connect

THE PROJECT ESSENTIALS

Raspberry Pi with Wi-Fi support

Ethernet cable

An active VPN subscription that supports OpenVPN

Secondary device such as a laptop to test your AP works

01 Choose your VPN

In order to carry out this project, you need an account with a VPN provider. Find a provider that supports the OpenVPN protocol, as this connection is generally considered to be the most secure. Free providers won't require any billing information but they are not as fast or reliable as paid services. If you choose a paid provider, try to find one that accepts anonymous payment methods such as Bitcoin. The website www.weusecoins.com has a list of these.

02 Download VPN configuration files

Attach the Pi to your router. Open Terminal or connect via SSH. If your VPN provider supports the secure OpenVPN standard, then they will have provided a configuration file with the extension .conf or .ovpn. For this tutorial a VPN configuration file from free provider VPNBook was used. Download the file to your Pi either by clicking the link or using wget in Terminal, for instance:

```
wget http://www.vpnbook.com/free-openvpn-account/VPNBook.com-OpenVPN-Euro1.zip
```

03 Install OpenVPN

The Raspbian repositories contain the OpenVPN software but not the most current version. Use the command su to switch to the root user then run these commands:

```
wget -O - https://swupdate.openvpn.net/repos/repo-public.gpg|apt-key add -
```

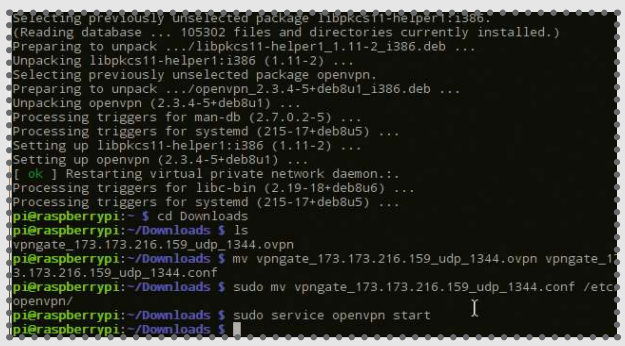
```
echo “deb http://swupdate.openvpn.net/apt jessie main” > /etc/apt/sources.list.d/swupdate.openvpn.net.list
```

```
apt-get update
```

```
apt-get install openvpn
```

Run **openvpn –version** to double-check you have the most up-to-date version of the software. At the time of writing this is 2.3.14.

04 Configure and run OpenVPN



Use the mv command to move the configuration file into the openvpn folder /etc/openvpn, amending the extension if necessary, for instance:

```
sudo mv vpngate_vpn151111650.opengw.net_udp_1344.ovpn /etc/openvpn/vpn1.conf
```

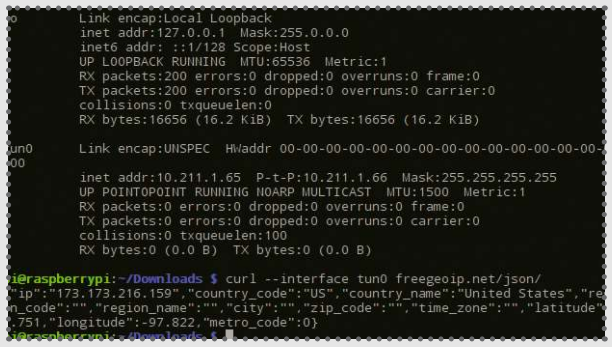
Next, start the OpenVPN service with the command sudo service openvpn start. Start OpenVPN using your .conf file with the openvpn –-config command, for instance:

```
sudo openvpn -config /etc/openvpn/vpn1.conf
```

Next, run

```
sudo service openvpn start
```

05 Test OpenVPN connection



Once the OpenVPN service is running, open a new tab in your Terminal or start a new SSH service and run the command ifconfig to list your network interfaces. Usually the VPN connection will appear as tun0. You can check your apparent location with the command:

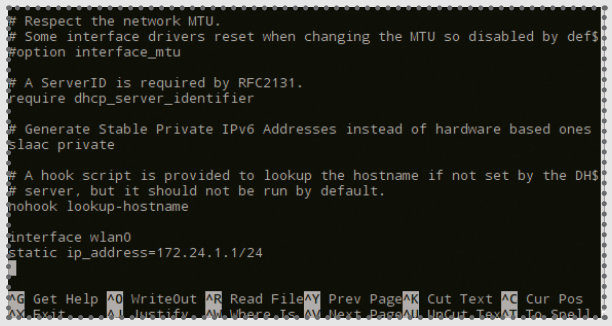
```
curl --interface tun0 freegeoip.net/json/
```

Next, make sure the OpenVPN service starts each time you log in. Then run

```
sudo nano /etc/default/openvpn
```

and remove the # at the start of the line reading "#AUTOSTART="all".

06 Install prerequisites



Now we'll install the necessary software to set up a Wireless AP. Do this by running

```
sudo apt-get install dnsmasq hostapd
```

Next, run

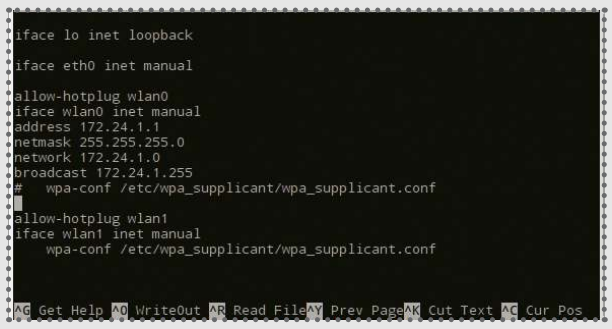
```
sudo nano /etc/dhcpcd.conf
```

Add these lines to the very bottom of the file:

```
interface wlan0
static ip_address=172.24.1.1/24
```

Press Ctrl+X, Y then Return to save and exit.

07 Set static IP



Open your network interfaces configuration with

```
sudo nano /etc/network/interfaces///ENDCODE
```

Change the line "iface wlan0 inet static" to "iface wlan0 inet manual". Press Return to start a new line, and then paste:

```
address 172.24.1.1
netmask 255.255.255.0
```

Automatic Login

If you need a username and password for your VPN, you can save these so OpenVPN will connect automatically. First run

```
sudo nano auth.txt
```

On the first line put the username and on the second put your password. Save and exit. Next edit your OpenVPN config file e.g.

```
sudo nano /etc/openvpn/vpn2.conf'
```

Scroll down to the line with the text "auth-user-pass". Leave a space and enter the path auth. ▶

```
network 172.24.1.0
broadcast 172.24.1.255
```

Place a '#' at the start of the line beginning "wpa-conf". Save and exit in the same way as before. Restart the dhcpcd service with

```
sudo service dhcpcd restart
```

08 Configure AP

Run

```
sudo nano /etc/hostapd/hostapd.conf'
```

and paste the following:

```
interface=wlan0
driver=nl80211
ssid=piVPN
hw_mode=g
channel=1
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=raspberry231
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Automatic Login

(cont.)

txt, for example:

```
auth-user-pass
/home/pi/auth.
txt
```

Save and exit once again.

FIND

MORE

FREE

MAGAZINES

FREEMAGS.CC

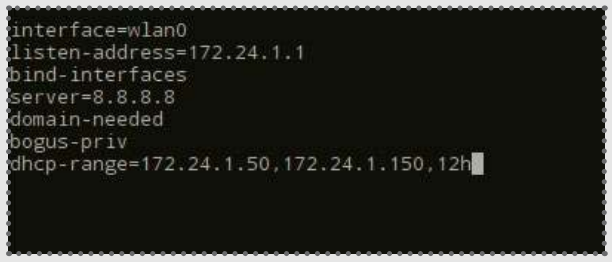
Feel free to change the name of the network from 'PiVPN' to one that is meaningful to you. Similarly change the password 'raspberry231' to something more secure.

Next run **nano /etc/default/hostapd**

Find the line starting #DAEMON_CONF="" and change to DAEMON_CONF="/etc/hostapd/hostapd.conf. Note the '#' at the start of the line must be removed.

09 Configure dnsmasq

Move the old dnsmasq configuration file with




```
sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
```

then create a new one by running

```
sudo nano /etc/dnsmasq.conf
```

Paste in the following text:

```
interface=wlan0
listen-address=172.24.1.1
bind-interfaces
server=8.8.8.8
domain-needed
bogus-priv
```

```
dhcp-range=172.24.1.50,172.24.1.150,12h
```

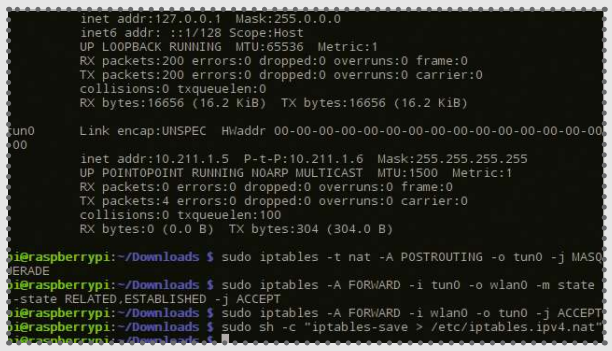
Note we are using Google's DNS server (8.8.8.8) for now; change this if you wish, then save and exit. Run

```
sudo nano /etc/sysctl.conf
```

Find the line starting "net.ipv4.ip_forward=1" and remove the '#' at the start. Now reboot the Pi.

10 Set up IPV4 Forwarding

For the next step, you need to run each of these commands individually:




```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

```
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

```
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

Next, run

> "We are using Google's DNS server (8.8.8.8) for now; change this if you wish, then save and exit"

```
sudo nano /etc/rc.local
```

Paste the following right above the line reading "exit 0":

```
iptables-restore < /etc/iptables.ipv4.nat
/usr/sbin/hostapd /etc/hostapd/hostapd.
conf
```

11 Test Access Point

Run the commands


```
sudo update-rc.d
hostapd enable
```

and

```
sudo update-rc.d dnsmasq enable
```

Reboot the Pi. You'll need a second device at this stage to see if you can access the Wireless AP. Search for it in your network menu and enter the password you created earlier on. If you can't remember this, run

```
sudo nano /etc/hostapd/hostapd.conf
```

on the Pi to view it again. Once connected visit **www.whatismyipaddress.com** to check you're behind the VPN.

12 Fix DNS Leaks

Certain VPN providers use their own DNS servers. Other VPN Providers are less cautious.

Visit **https://www.dnsleaktest.com/** and click Extended Test to check you're safe. If any of the DNS servers match your regular ISP, your connection is not fully secure. To resolve this, edit your VPN configuration file e.g.

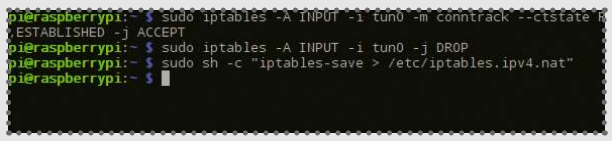

```
sudo nano /etc/openvpn/vpn2.conf
```

and add these lines immediately above "<ca>":

```
script-security 2
up /etc/openvpn/update-resolv-conf
down /etc/openvpn/update-resolv-conf
```

Save and exit, then restart your Pi. Check the DNS leak website once again. If this fails to resolve the issue, try using another VPN provider.

13 Block unsolicited connections



As your Pi is sitting between your computer and the internet, it can potentially be accessed by other devices.

Prevent unsolicited incoming connections from other devices with the following commands:

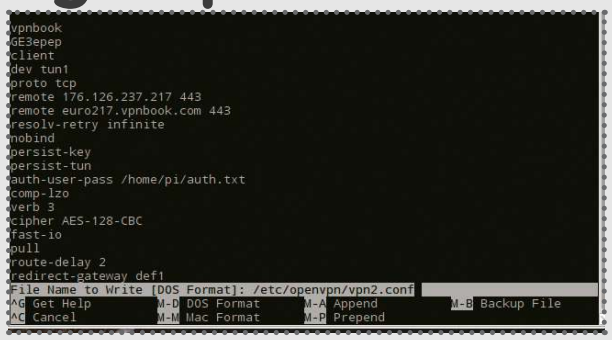
```
sudo iptables -A INPUT -i tun0 -m
conntrack --ctstate RELATED,ESTABLISHED -j
ACCEPT
```

```
sudo iptables -A INPUT -i tun0 -j DROP
```

Make sure to save your changes so that they'll apply on reboot:

```
sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

14 Route all traffic through OpenVPN



When you boot the Pi initially, certain applications may try to connect directly to the internet, which can undermine your anonymity.
To channel all network traffic through the VPN, you need to edit your configuration file in /etc/openvpn, for instance by running

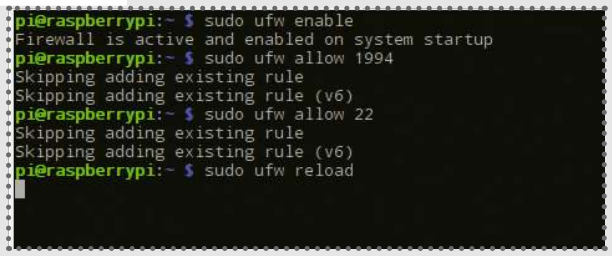
```
sudo nano /etc/openvpn vpn2.conf
```

Make sure the line "redirect-gateway" reads "redirect gateway def1". DNS queries will also be routed through the VPN also so make sure your provider supports this.

15 Set up Firewall

Although you may have previously configured iptables to prevent unsolicited incoming connections, to be on the safe side, consider installing ufw (Uncomplicated Firewall) with

```
sudo apt-get install ufw
```



Run

```
sudo ufw enable
```

to fire it up, then open the default OpenVPN port 1194 with

```
sudo ufw allow 1194
```

You may also want to enable Port 22 to allow connecting via SSH. Remember that this doesn't change which ports are open and closed on the router.

Draw 3D text in Minecraft using Python code

In past issues, we've hooked directly into Minecraft with Python code. This time we're using Turtle Graphics to draw 3D text

Minecraft's 'Creative Mode' is indeed a great mode to get creative with, but it's a time-consuming job placing blocks down into your world one-by-one. That's why we put together a package which incorporates a range of modifications that enable us to execute Python scripts directly into a Minecraft world. McPiFoMo includes MCPiPy by 'fleap' and 'bluepillRabbit' of **https://mcpipy.wordpress.com**, and the Raspberry Jam Mod, developed by Alexander Pruss. We'll also be using Minecraft Turtle, which was put together by Martin O'Hanlon of **http://stuffaboutcode.com**.

That's a lot of mods, but together they allow us to do with Minecraft on Linux what would usually have only been possible with the Raspberry Pi edition of Minecraft.

As a prerequisite, we assume you've installed McPiFoMo from the link in the sidebar.

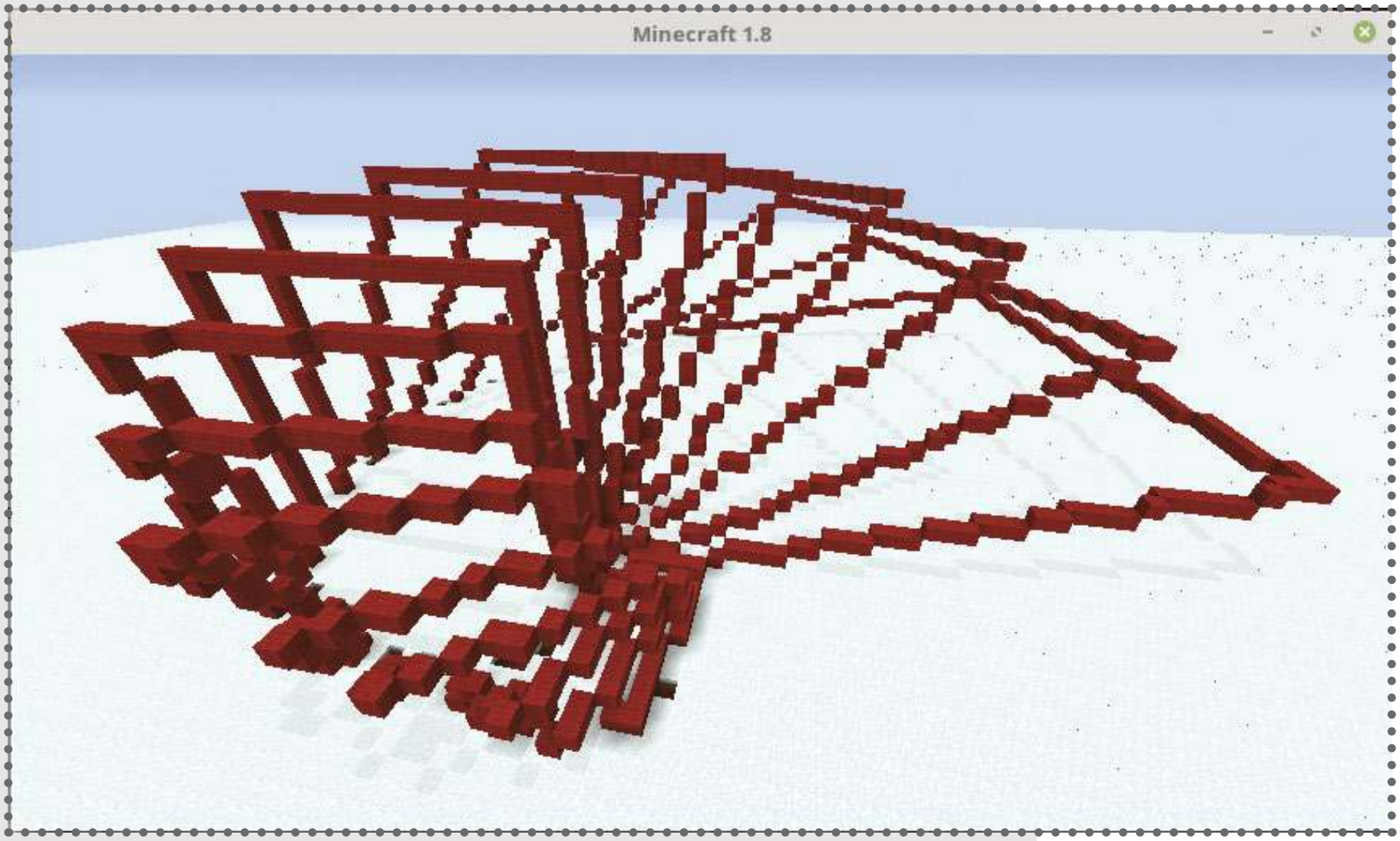
THE PROJECT ESSENTIALS

Minecraft (www.mojang.com/games)

Python (www.python.org)

McPiFoMo (http://rogerthat.co.uk/McPiFoMo.rar)

Minecraft Turtle (http://bit.ly/MinecraftTurtle)



01 Minecraft Turtle Library

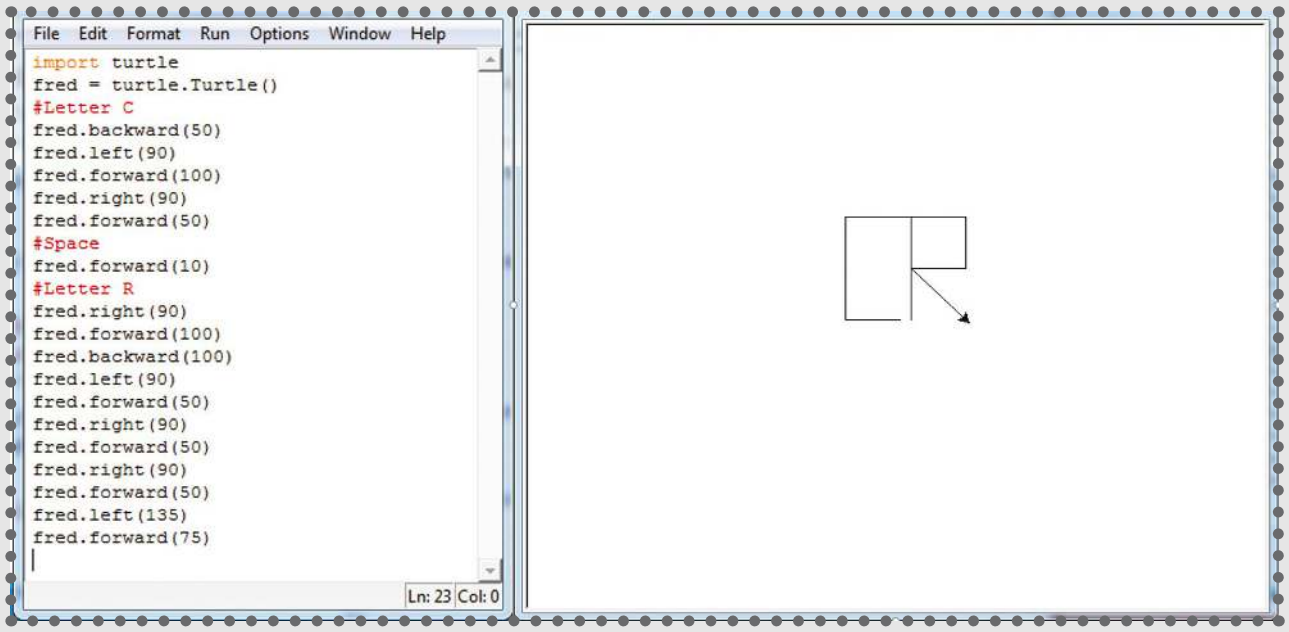
Turtle Graphics are bundled with Python, but to get this kind of functionality in Minecraft we'll need to download a custom Minecraft Turtle Library:

```
cd ~
git clone https://github.com/
martinohanlon/minecraft-turtle.git
```

To install it:

```
cd ~/minecraft-turtle
python setup.py install
python3 setup.py install
```

If you've used Turtle Graphics before, you'll have no



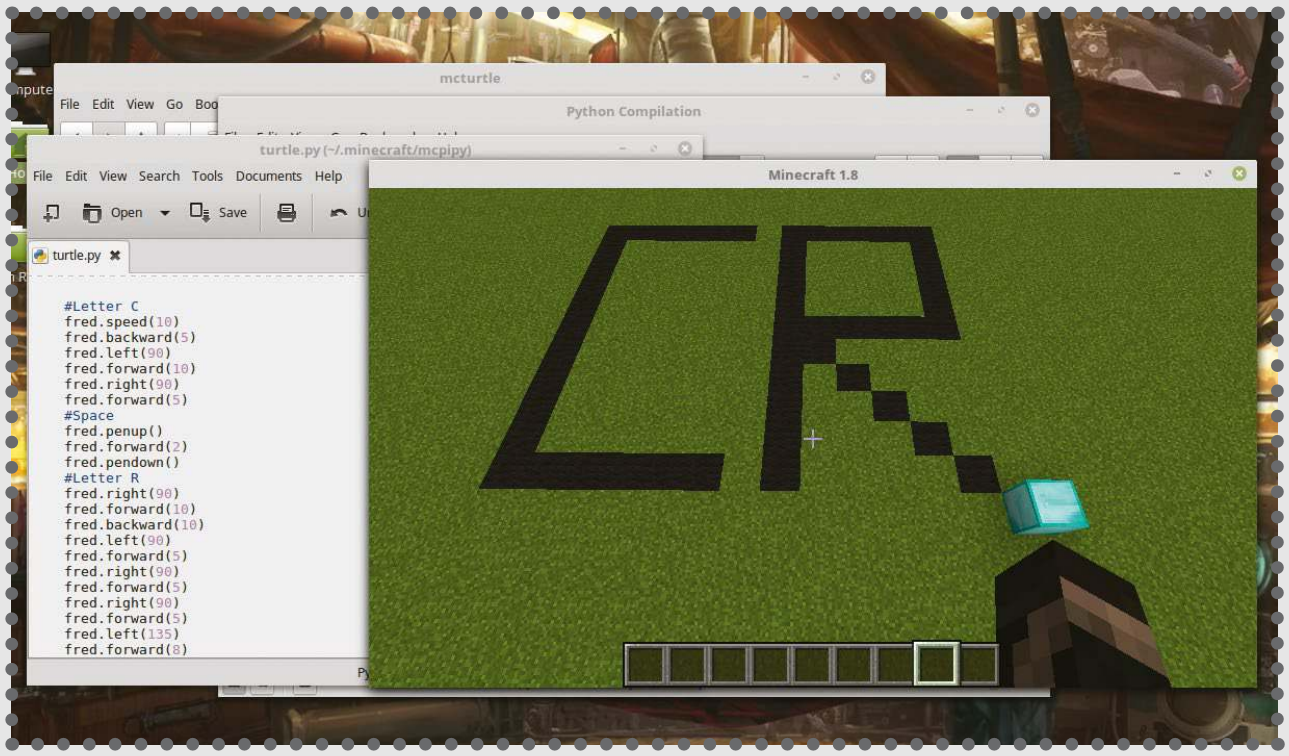
problem getting to grips with this incarnation.

02 Getting to know Turtle Graphics

Imagine you stuck a paintbrush in the mouth of a real turtle; everywhere that turtle moves, it drags the brush along, drawing a path. That's the basis of Turtle Graphics. You're drawing vector graphics with a relative cursor, across a virtual canvas. We direct the turtle where to go, and it leaves a trail of lines behind it. In Minecraft, those lines are represented by blocks.

03 Third dimension

Traditionally, Turtle Graphics programs can move



Action	Command	Example
Walk forward:	turtlename.forward(distance)	fred.forward(100)
Walk backward:	turtlename.backward(distance)	fred.backward(100)
Rotate left:	turtlename.left(angle)	fred.left(45)
Rotate right:	turtlename.right(angle)	fred.right(90)
Move up:	turtlename.up(distance)	fred.up(100)
Move down:	turtlename.down(distance)	fred.down(100)
Stop drawing:	turtlename.penup()	fred.penup()
Start drawing:	turtlename.pendown()	fred.pendown()

forward and backward, but not necessarily left or right. We instead rotate left/right in degrees, and move forward/backward to draw our lines in the direction we need. Minecraft Turtle, however, has an additional dimension, with up and down commands moving toward to the sky/ground accordingly. Pictured above is a list of the basic commands we'll be using to traverse our virtual canvas.

04 Test your turtle

Create a new Python script in the IDLE and input:

```
from mcturtle import minecraftturtle
from mcpi import minecraft
from mcpi import block

#Connect to Minecraft and find the
player's current position
mc = minecraft.Minecraft.create()
pos = mc.player.getPos()

#Spawn a new turtle, giving the variable
```


```
a name of your choice
fred = minecraftturtle.MinecraftTurtle(mc,
pos)

#Test your turtle
fred.forward(10)
fred.backward(20)
```

05 Draw your initials

Now we're going to draw out our initials. Some trial and error will be needed. Here's how we drew 'CR':

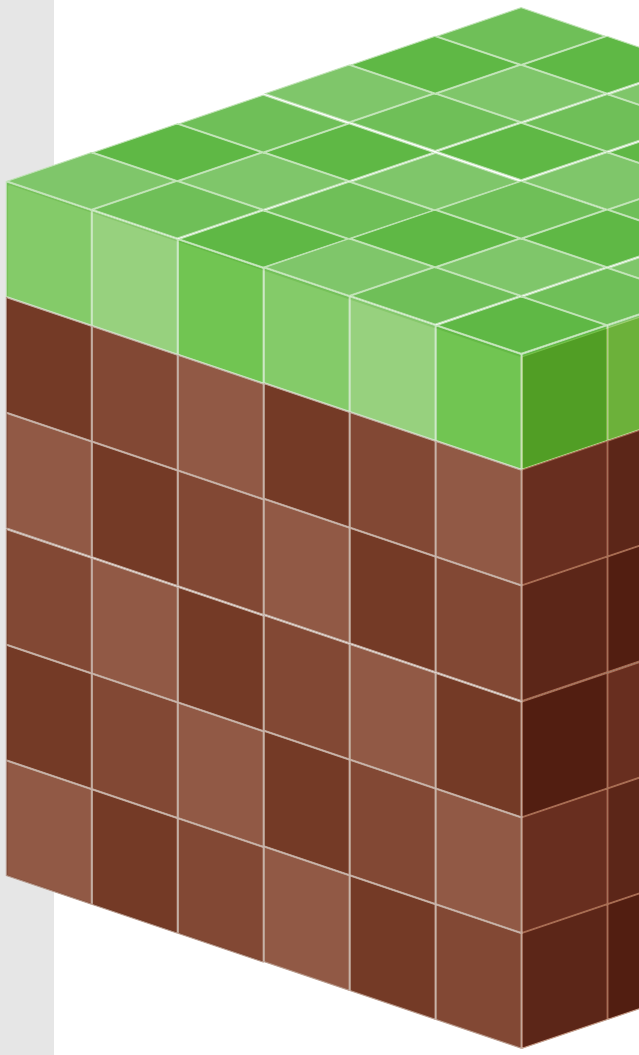
```
#Draw a sharp letter C
fred.backward(50)
fred.left(90)
fred.forward(100)
```


```
fred.right(90)
fred.forward(50)

#Provide a little space, by picking up
the pen and moving along
fred.penup()
fred.forward(30)
fred.pendown()

#Draw the letter R
fred.right(90)
fred.forward(100)
fred.backward(100)
fred.left(90)
fred.forward(50)
```

```
fred.right(90)
fred.forward(50)
fred.right(90)
fred.forward(50)
fred.left(135)
fred.forward(75)
```

06 Spruce up your turtle

You'll likely want to change the type of block your turtle is drawing with: fred.penblock(block.DIRT.id) – where DIRT is a Minecraft block type; others include WOOD, GRASS, WOOL, TNT and DIAMOND. To check if your pen (or paintbrush) is up or down at any given time, use: print fred.isdown(). You can also change the speed of your turtle, 1 being a turtle, 10 being a hare: fred.speed(10).



07 Position your 3D initials

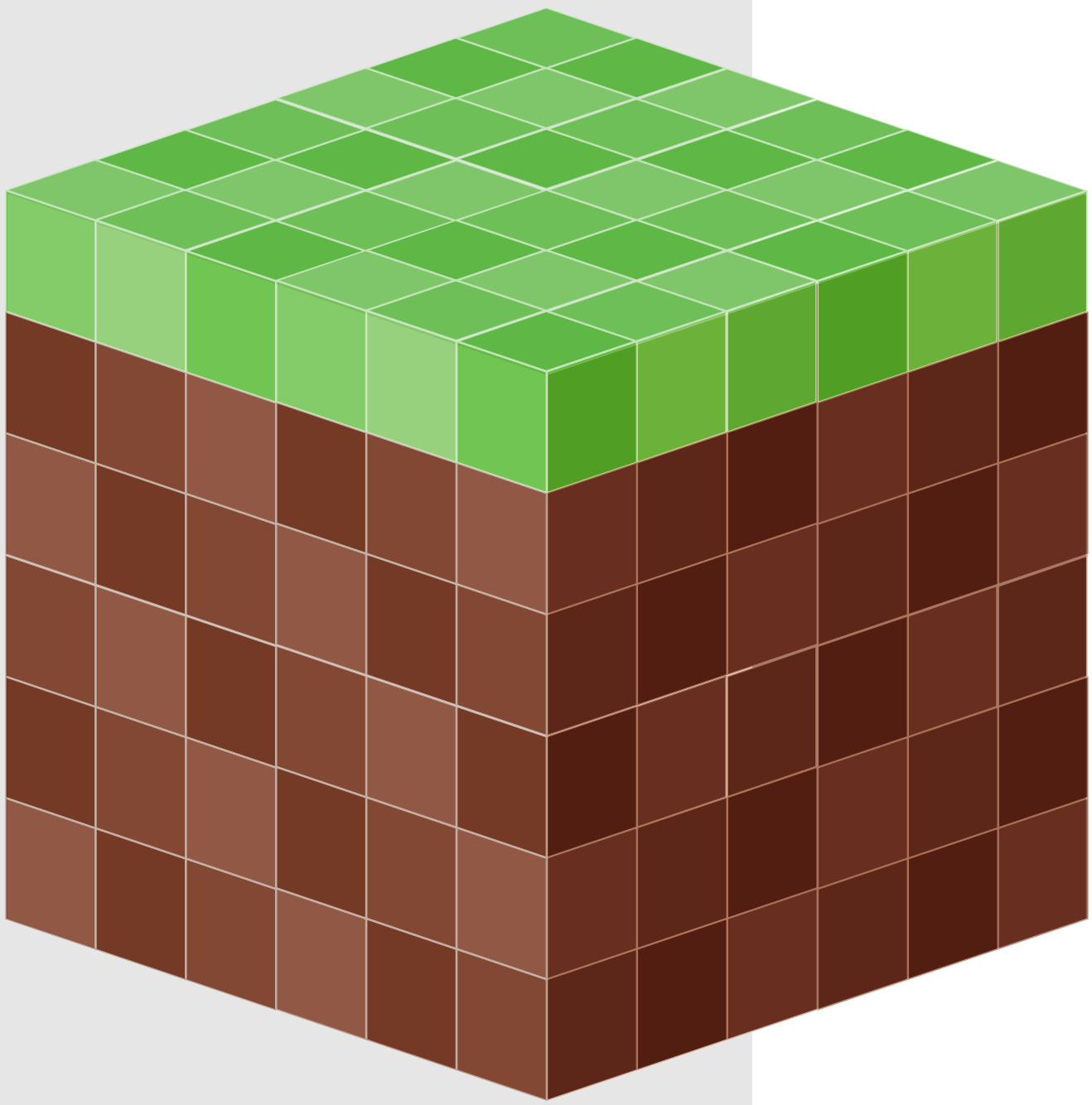
You may want to change the location of your turtle, before or during the drawing process. To do this, we can use print or set commands to alter the position with Minecraft coordinates (x,y,z), just as you would to teleport another player around your world.

Pairing these commands with penup() and pendown() allows you to keep your writing neat, kerning your letters when necessary.

```
#Print your turtle's position
turtlePos = fred.position
print(turtlePos.x)
print(turtlePos.y)
print(turtlePos.z)
```

```
#Reassign your turtle's position
fred.setposition(0,0,0)
fred.setx(0)
fred.sety(0)
fred.setz(0)
```

There's also a handy shortcut to send your turtle immediately back to the location it started in: fred.home().



Preserve your Raspberry Pi's Micro SD card

Use a few tweaks and USB-booting to prolong your SD card and avoid losing important Raspberry Pi projects

At some point during the lifetime of your Pi, you are likely to encounter a problem with your SD/microSD card. If you're lucky, this will be minimal; perhaps a reboot will fix it.

If you're unlucky, it could mean the end of your SD card, and the loss of all data on your Pi. The simple fact is that SD cards do not last forever. Flash storage, by design, has a limited number of read/write cycles. While error-correction software is built-in, and cards ship with larger-than-described storage to cover damaged blocks, eventually corruption will cause a problem.

SD card corruption occurs in various ways. It might be due to a sudden variation in voltage during a read/write cycle or from being removed from the Raspberry Pi. Flash storage is also susceptible to extreme temperatures and physical damage. Cheap SD cards, meanwhile, are usually unreliable; whatever the situation, you should rely on the more expensive cards from SanDisk or Kingston.

While it's useful to regularly back up your flash storage to enable quick recovery, a more proactive option is to bypass the SD card entirely by relying on other storage for booting the Raspberry Pi, but you should also be aware of various tools that can be used to protect your microSD card.

THE PROJECT ESSENTIALS

USB storage device

Ethernet cable

01 Don't flash a fresh OS for every project

The read/write cycle limit is a physical hardware restriction preventing infinite reuse of an SD card. So look after it! While 'swappiness' (the kernel parameter defining how much and often RAM is copied to storage) is apparently set low in Raspbian, there is more you can do.

One way to extend the lifespan of the device is to avoid flashing a fresh version of Raspbian (or whatever your

```
pi@raspberrypi:~ $ sudo apt-get remove dnsmasq
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  dns-root-data dnsmasq-base libmnl0 libnetfilter-conntrack3
Use 'apt-get autoremove' to remove them.
The following packages will be REMOVED:
  dnsmasq
0 upgraded, 0 newly installed, 1 to remove and 71 not upgraded.
After this operation, 117 kB disk space will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 31388 files and directories currently installed.)
Removing dnsmasq (2.72-3+deb8u1) ...
pi@raspberrypi:~ $
```

preferred OS is) every single time you start a new project.

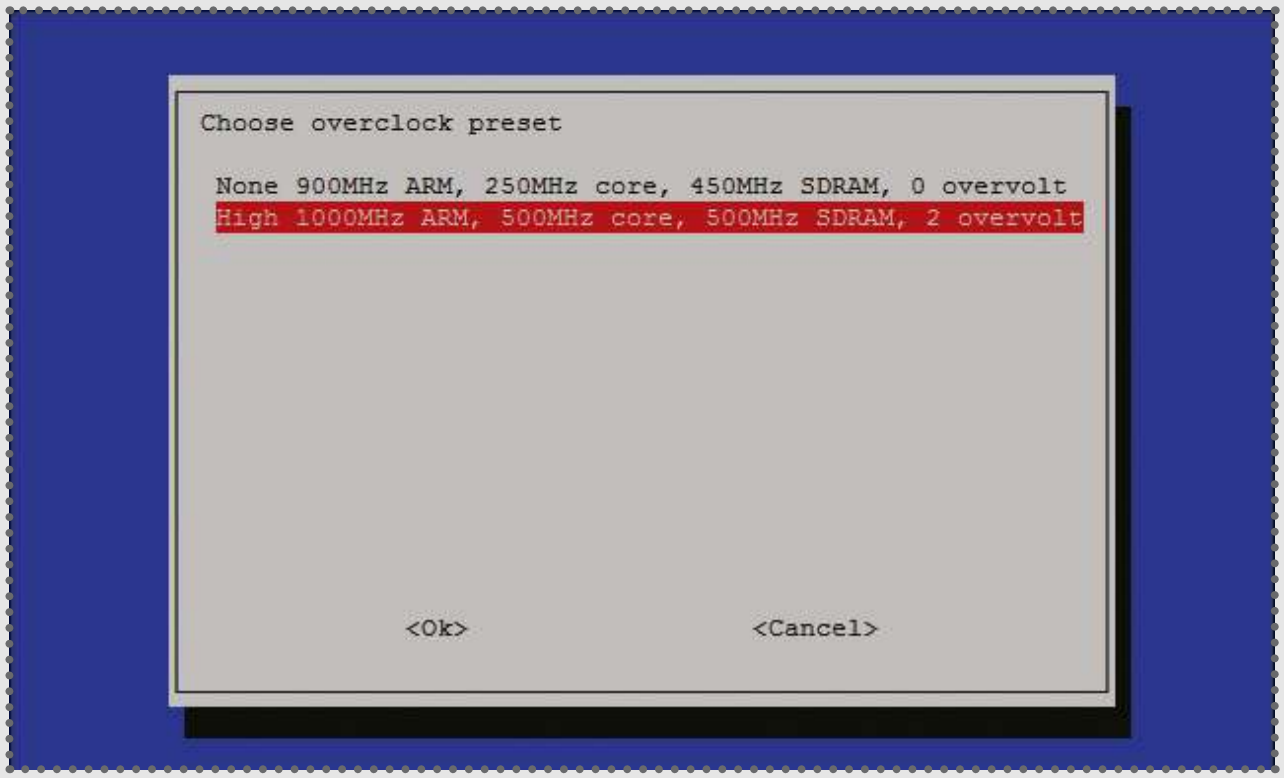
Doing so applies a card-wide reduction in the remaining read-write cycles. So, by maintaining a working microSD card throughout several projects, you avoid the effect that regular flashing has on the card.

This is not a perfect solution, but it will help you get projects started without the initial flashing and configuration that is typically required. Need to keep things tidy? Remove software with sudo apt-get remove APPNAME. This will uninstall the software you no longer need, but may be time-consuming. In short, only flash Raspbian when you really must.

02 Maintain a constant power supply

A reliable power supply is a major aspect in the preservation of your Raspberry Pi's SD storage. If there's much variation, data can be lost, ultimately causing corruption of the card. At this point, your Pi probably won't boot, and a new OS will need flashing.

The Pi requires a constant voltage of 5V. This is available via the approved mains adaptors, but keep in mind that these devices cannot account for failings in the sockets they're plugged into. Don't use cheap extension leads. Instead, ensure your mains adaptor is connected either to the wall (if you have modern surge protection built in) or



otherwise to an extension with surge protection.

It's common to attempt to squeeze as much power out of a Raspberry Pi through overclocking, but this too is a possible cause of disk corruption. Rather than go through the rigmarole – and card-degrading – act of flashing a new ROM, it's safer to avoid overclocking the Pi's CPU if you want to maintain a stable system.

03 Write to RAM, not storage

A great way to reduce the number of read/write cycles on your SD card is to not write to it in the first place. This doesn't mean leaving your Pi powered off! Instead, you can write everything to the computer's RAM. As such, nothing will be written to the microSD card, thereby extending its life.

Better still, this is easy to set up using tempfs. Begin by opening the fstab in nano with sudo nano /etc/fstab.

At the bottom of the file, add this line:

```
tmpfs/var/log tmpfs defaults,noatime,nosuid,
oe=0755,size=100m 0 0
```

Press Ctrl+X to exit and save. This moves the /var/log directory to RAM, reducing the microSD card's read/write

Avoid SD card corruption with checks

It is possible to run checks to give you advance warning of impending SD card corruption. Begin with this command to check the filesystem every x number of reboots. Change [x] for your preferred number: **tune2fs -c [x] /dev/mmcblkp02**. You might also enable autocorrection, to help with accurate data storage with **nano /etc/default/rcS --> FSCKFIX=yes.**

```
 GNU nano 2.2.6               File: /etc/fstab                        Modified

proc            /proc           proc    defaults          0       0
/dev/mmcblk0p1  /boot           vfat    defaults          0       2
/dev/mmcblk0p2  /               ext4    defaults,noatime  0       1
# a swapfile is not a swap partition, no line here
#   use  dphys-swapfile swap[on|off]  for that
tmpfs /var/log tmpfs defaults,noatime,nosuid,mode=0755,size=100m 0 0
```

cycles. Other locations can be safely moved to RAM too:

```
tmpfs /tmp tmpfs defaults,noatime,nosuid,siz
=100m 0 0
tmpfs /var/tmp tmpfs defaults,noatime,nosuid
size=30m 0 0
tmpfs/var/log tmpfsdefaults,noatime,nosuid,mo
de=0755,size=100m 0 0
tmpfs /var/run tmpfs defaults,noatime,nosuid,
mode=0755,size=2m 0 0 tmpfs /var/spool/mqueue
tmpfs defaults,noatime,nosuid,mode=0700,gid=12
,s
ize=30m 0 0
```

> “SD cards do not last forever. Flash storage has a limited number of read/write cycles”

Beware: this will only last until you reboot your Pi, at which point everything is cleared from RAM.

04 Boot your Pi from a USB stick

Recent updates to Raspbian enable you to boot a Raspberry Pi 3 via an attached USB device (a flash drive, standard HDD or even an SSD), bypassing the microSD card entirely. Via SSH, begin with:

```
sudo apt-get update
sudo BRANCH=next rpi-update
```

Then enable USB boot mode:

```
echo program_usb_boot_mode=1 | sudo tee -a
boot/config.txt
```

```
pi@raspberrypi: ~
File Edit View Search Terminal Help
pi@raspberrypi:~ $ vcgencmd otp_dump | grep 17
17:3020000a
pi@raspberrypi:~ $
```

With the program_usb_boot_mode=1 instruction added to the end of the config.txt file, reboot with sudo reboot.

When the Pi restarts, you'll need to check if the one-time programmable (OTP) memory has been changed.

```
vcgencmd otp_dump | grep 17:
```

If the previous steps have been successful, you should see something like '0x3020000a' (pictured, above). Your Raspberry Pi is now ready to boot from a USB device, so connect the one that you want to use; but note that it will be reformatted. You can Identify your USB device with lsblk, which will list all block devices.

Connected USB devices are usually called 'sda'. Enter the following to unmount the device and run the Parted tool:

```
sudo umount /dev/sda
```

```
pi@raspberrypi: ~
File Edit View Search Terminal Help
File system type?  [ext2]? ^Z
[1]+  Stopped                 sudo parted /dev/sda
pi@raspberrypi:~ $ sudo parted /dev/sda
GNU Parted 3.2
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mktable msdos
Warning: The existing disk label on /dev/sda will be destroyed and all data on
this disk will be lost. Do you want to continue?
Yes/No? yes
(parted) mkpart primary fat32 0% 100M
(parted) mkpart primary ext4 100M 100%
(parted) print
Model: SanDisk Cruzer Fit (scsi)
Disk /dev/sda: 32.0GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  99.6MB  98.6MB  primary  fat32        lba
 2      99.6MB  32.0GB  31.9GB  primary  ext4         lba

(parted)
```

```
sudo parted /dev/sda
```

This will take you to the Parted prompt.

05 Copy Raspbian to your USB drive

At the prompt, enter:

```
mkpart primary fat32 0% 100M
mkpart primary ext4 100M 100%
print
```

Then use Ctrl+C to exit. Back at the command prompt, you will need to create both a new boot filesystem and root filesystem:

```
sudo mkfs.vfat -n BOOT -F 32 /dev/sda1
sudo mkfs.ext4 /dev/sda2
```

Next, mount the target filesystems:

```
sudo mkdir /mnt/target
sudo mount /dev/sda2 /mnt/target/
sudo mkdir /mnt/target/boot
sudo mount /dev/sda1 /mnt/target/boot/
sudo apt-get update; sudo apt-get install
rsync
```

To copy Raspbian to your USB device use:

```
sudo rsync -ax --progress / /boot /mnt
target
```

This will take a while to complete, so leave it to finish. Once done, you'll need to copy the SSH host keys from the microSD

Get the best value SD cards

Low-cost SD and micro SD cards are generally unreliable. But how can you tell what's good quality or not? First, check the brand. Buy cards from SanDisk or Kingston as a rule. Second, look at the Speed Class rating. SDHC and SDXC cards are available, fast, reliable media. The higher the speed rating (upwards from 2MB/sec), the better the card.

card to the USB device to maintain the connection via SSH. Enter the following a line at a time.

```
cd /mnt/target
sudo mount --bind /dev dev
sudo mount --bind /sys sys
sudo mount --bind /proc proc
sudo chroot /mnt/target
rm /etc/ssh/ssh_host*
dpkg-reconfigure openssh-server
exit
sudo umount dev
sudo umount sys
sudo umount proc
```

06 Prepare to boot from USB

Before you reboot your Pi from the USB device, edit the cmdline.txt file again in the terminal:

```
sudo sed -i "s,root=/dev/mmcblk0p2,root=/dev
sda2," /mnt/target/boot/cmdline.txt
```

A similar change must also be made to /etc/fstab:

```
sudo sed -i "s,/dev/mmcblk0p,/dev/sda," /mnt
target/etc/fstab
```

You're now ready to unmount the filesystem:

```
cd ~
sudo umount /mnt/target/boot
sudo umount /mnt/target
```

"You might retain the Raspbian image to prepare other Pis for USB booting in future"

```
pi@raspberrypi: /mnt/target
File  Edit  View  Search  Terminal  Help
pi@raspberrypi:/mnt/target $ sudo sed -i "s,root=/dev/mmcblk0p2,root=/dev/sda2,"
 /mnt/target/boot/cmdline.txt
pi@raspberrypi:/mnt/target $ sudo sed -i "s,/dev/mmcblk0p,/dev/sda," /mnt/target
/etc/fstab
pi@raspberrypi:/mnt/target $
```

At this stage, you can enter the poweroff command to shut down your Pi. Once the lights are off, disconnect the power supply and remove the microSD card. A few minutes later, you can reconnect the power and boot your Pi – from the USB device!

Your microSD card is now guaranteed a much longer lifespan. You might retain the Raspbian image, however, for preparing other Pis for USB booting in future.

Better still, this means that you can have as much storage as possible for your Raspberry Pi. USB flash usually stops around the 512GB capacity (although there are larger devices) while mechanical hard disk drives can potentially add terabytes of storage to your Pi. Alternatively, an SSD device will speed things up considerably.

07 Boot Raspbian across your network!

Booting from USB can be taken to the next level – network boot. Using a Pi as a server, you can set up a Raspberry Pi 3 with Raspbian Lite and set it to initially boot from USB. With the server and new client configured correctly, the Pi can boot from the network, again reducing the impact on the SD card.

On your intended client, follow the previous steps up to the point of removing program_usb_boot_mode from /boot/config.txt, then running the poweroff command.

Next, remove the SD card and insert it into the Pi you'll be using as a server. Boot this device, then run sudo raspi-config. This will open the configuration options. Select the

```
        140,095 100%  422.26kB/s    0:00:00 (xfr#27431, to-chk=13/35841)
var/log/wtmp
          9,216 100%   27.52kB/s    0:00:00 (xfr#27432, to-chk=12/35841)
var/log/apt/
var/log/apt/history.log
         11,369 100%   33.75kB/s    0:00:00 (xfr#27433, to-chk=7/35841)
var/log/apt/term.log
         91,006 100%  263.72kB/s    0:00:00 (xfr#27434, to-chk=6/35841)
var/log/fsck/
var/log/fsck/checkfs
             31 100%    0.09kB/s    0:00:00 (xfr#27435, to-chk=5/35841)
var/log/fsck/checkroot
             31 100%    0.09kB/s    0:00:00 (xfr#27436, to-chk=4/35841)
var/log/ntpstats/
var/log/samba/
var/mail/
var/opt/
var/spool/
var/spool/mail -> ../mail
var/spool/cron/
var/spool/cron/crontabs/
var/spool/rsyslog/
var/tmp/
pi@raspberrypi:~ $
```

Expand Filesystem option. Then create a copy of the root filesystem:

```
sudo mkdir -p /nfs/client1
sudo apt-get install rsync
sudo rsync -xa --progress --exclude /nfs /nfs/client1
```

This will take a while to complete, so be patient!

```
pi@raspberrypi:~ $ ip route | grep default | awk '{print $3}'
192.168.0.1
pi@raspberrypi:~ $ ip -4 addr show dev eth0 | grep inet
    inet 192.168.0.40/24 brd 192.168.0.255 scope global eth0
pi@raspberrypi:~ $
```

08 Find the addresses

Continue by maintaining your connection via SSH, by regenerating the SSH host keys:

```
cd /nfs/client1
sudo mount --bind /dev dev
sudo mount --bind /sys sys
sudo mount --bind /proc proc
```

```
sudo chroot .
rm /etc/ssh/ssh_host_*
dpkg-reconfigure openssh-server
exit
sudo umount dev
sudo umount sys
sudo umount proc
```

Then find the address of your router. If you don't know this, run:

```
ip route | grep default | awk '{print $3}'
```

Check your Pi's own IP address with:

```
ip -4 addr show dev eth0 | grep inet
```

Then use cat /etc/resolv.conf to find the address of your DNS server. You should now have your device's IP, broadcast and DNS server addresses, so note these down. Run sudo nano /etc/network/interfaces and edit the line iface eth0 inet manual so that it reads:

```
auto eth0
iface eth0 inet static
address [YOUR_IP_ADDRESS]
netmask 255.255.255.0
gateway [YOUR_BROADCAST_ADDRESS]
```

Press Ctrl+X to save and exit. To make this work, you'll need to disable DHCP networking, so run the following and then reboot your Pi:

```
sudo systemctl disable dhcpcd
sudo systemctl enable networking
```

09 Configure your server's network settings

Enter the following with the DNS IP address you noted earlier.

```
echo "nameserver [YOUR_NAMESERVER_IP]" | sudo tee -a /etc/resolv.conf
```

Prevent changes to this with:

```
sudo chattr +i /etc/resolv.conf
```

You then need to install some software:

```
sudo apt-get update
sudo apt-get install dnsmasq tcpdump
```

The dnsmasq tool can cause problems, so prevent this with:

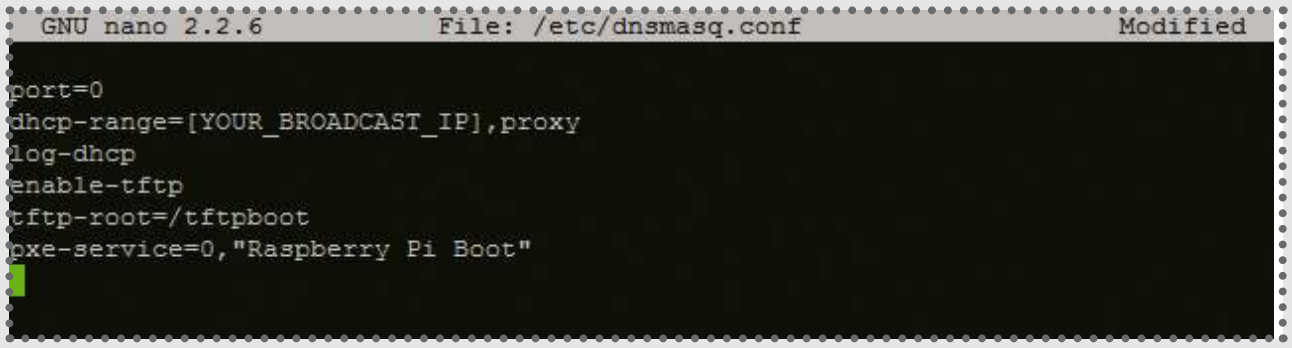
```
sudo rm /etc/resolvconf/update.d/dnsmasq
```

Reboot again, then run tcpdump to detect DHCP from the client Pi.

```
sudo tcpdump -i eth0 port bootpc -v
```

At this stage, connect the client Pi to the network via Ethernet, and connect the power cable. After ten seconds, the LEDs should light, and a packet from the client will be received by the server and displayed in the tcpdump tool. Press Ctrl+C to exit, then enter

```
sudo echo | sudo tee /etc/dnsmasq.conf
```




```
sudo nano /etc/dnsmasq.conf
```

Delete everything in the file and add:

```
port=0
dhcp-range=[YOUR_BROADCAST_IP],proxy
log-dhcp
enable-tftp
tftp-root=/tftpboot
pxe-service=0,"Raspberry Pi Boot"
```

Next, create a new directory, tftpboot:

```
sudo mkdir /tftpboot
sudo chmod 777 /tftpboot
sudo systemctl enable dnsmasq.service
sudo systemctl restart dnsmasq.service
```

10 Prepare for network boot

Continue by monitoring the dnsmasq log with:

```
tail -f /var/log/daemon.log
```

If working, a 'not found' message will be displayed. Copy the necessary files with cp -r /boot/* /tftpboot.

Then restart dnsmasq with sudo systemctl restart dnsmasq and the client Pi is ready to boot the root filesystem and then boot from the network. The /nfs/client1 filesystem must now be exported:

```
 GNU nano 2.2.6          File: /nfs/client1/etc/fstab

proc            /proc           proc    defaults          0       0
/dev/mmcblk0p1  /boot           vfat    defaults          0       2
/dev/mmcblk0p2  /               ext4    defaults,noatime  0       1
# a swapfile is not a swap partition, no line here
#   use  dphys-swapfile swap[on|off]  for that
```

```
sudo apt-get install nfs-kernel-server
echo "/nfs/client1 *(rw,sync,no_subtree
check,no_root_squash)" | sudo tee -a /etc/
exports
sudo systemctl enable rpcbind
sudo systemctl restart rpcbind
sudo systemctl enable nfs-kernel-server
sudo systemctl restart nfs-kernel-server
```

Next, edit /tftpboot/cmdline.txt, changing the line beginning 'root=' to:

```
root=/dev/nfs nfsroot=[YOUR_DEVICE_IP]:/nfs
client1 rw ip=dhcp rootwait elevator=deadline
```

Open fstab (with sudo nano /nfs/client1/etc/fstab) and remove the /dev/mmcblkp1 and /dev/mmcblkp2 lines. You're done! Now start the client and wait for it to boot from the network. This may be a little slower than what you're used to (depending on your network speed), but will extend the life of your Pi's microSD card considerably.

Anaconda for the Pi

Python is the language of choice on the Raspberry Pi, and there are many packages available within the Raspbian repository. Berryconda helps you move out beyond these modules

Python has always been the language of choice for projects on the Raspberry Pi. There are several reasons for this, but we won't dive into the history here. As a consequence of this decision, there are many Python modules available as packages within the Raspbian package repository. You can trust that these have been compiled to run on the ARM processor and that they have been tested and are therefore safe to use. But not all Python modules are available this way. For the missing modules, the intention is that you could use pip to install them yourself on your Raspberry Pi. While this works fine for some modules, you will eventually start running into those modules that require debugging to figure out why they aren't working correctly on the ARM processor. The amount of work involved in debugging these types of issues really should be distributed across many people, and luckily it is. For Python coders, a very good distribution of Python modules is available as the Anaconda project. There is also a community port of the Anaconda project, called Berryconda, which has been ported to the Raspberry Pi. The main website for the project is located at github.com/jjhelmus/berryconda. This article will cover the steps to getting it installed and ready to use on your Raspberry Pi so that we can go on and look at some of the

functionality that becomes available to you for Python coding in future articles.

The very first step is to download and install the core of the Berryconda system. On the download page, there are installation files for both armv6l (Raspberry Pi 1 or Zero) and armv7l (Raspberry Pi 2 or 3). There are also different versions of the installer for Python 2.X or Python 3.X. This covers all of the options that you might need for your particular project. The installer is actually a shell script, so all you need to do is to make the script executable, and then run it. For example, we can install the Python 3.x version on a Raspberry Pi 1 with the following commands:

```
wget https://github.com/jjhelmus/berryconda/releases/download/v1.0.0/Berryconda3-1.0.0-Linux-armv6l.sh
chmod +x Berryconda3-1.0.0-Linux-armv6l.sh
./Berryconda3-1.0.0-Linux-armv6l.sh
```

By default, this will install the base of an Anaconda environment into the directory berryconda3 in your home directory. You can change this installation directory during installation in case you want to put it somewhere else. The installer also asks you whether you want to add the path to the binaries to your PATH environment variable. This is generally a good idea. Just remember to exit from the current shell and log back in so that the new path is picked up. Either that, or manually source the initialisation file to get it set.

The main utility within Anaconda is the conda packaging system. Conda provides a fully featured package management system to handle Python modules and all of the various dependencies that may be required. There is a very good set of documentation available at the website

conda.io/docs, covering all of the options and functionality available. The very first thing you will want to do is to keep your current system up to date. You can update individual packages with the command:

```
conda update package-name
```

This command will go out to the internet and figure out what new versions of the given package exist, and if it finds one, it will ask you whether you really want to perform the update. If you want to just see what will be done, you can use the command line option --dry-run to get a display of what commands would be run. If you just want to keep the entire system updated, you can use the following command to handle updating everything:

```
conda update --all
```

To install new packages, you need to first find out what has already been packaged and is available. Because this is a community effort, the selection will vary over time. So you should check before trying to install some packages. You can do that search with the command:

```
conda search some_text
```

This will do a regular expression search of package titles, looking for the given text, and return a list of any available. Sometimes, however, you may get a rather large list returned. If this happens, you can do a more refined search using the usual regex options used in many other UNIX utilities. If you already know the package name, you could use the -f option to force the search to only return exact matches to the text you

give it. If the package in question is already installed on your Raspberry Pi, it will have an asterisk beside it for the version installed. When you are ready to install the package you were searching for, you can do so with the following command:

```
conda install ipython
```

This will, by default, install the latest version of the ipython package within the Berryconda environment. It will also install any missing dependencies, as well as updating any out-of-date dependencies.

While the online documentation is great, there are also help pages available within conda. If you want general help, you can get it with the help command. For help with some specific conda commands, you can use something like the following command:

```
conda install --help
```

This is handy for all of the details that nobody can seem to remember. Over time, you may find that not all of the installed packages are needed any more. This could be an issue on a small machine like the Raspberry Pi. The first step is to clean up the Berryconda environment. You can remove unused versions of packages, cached installation tarballs of the packages in the environment, and index caches. Assuming that you simply want as lean a system as possible, you can clean up the entire environment with the following command:

```
conda clean --all
```

If there are installed packages that are no longer needed, you can remove them with the uninstall command. For example,

"The first step is to clean up the Berryconda environment"

the following command would remove the scipy Python module:

```
conda uninstall scipy
```

If you have a drastic need to restart, you could remove everything from the environment by using the --all command option.

One of the great strengths of Python is the large set of third-party modules available for extending functionality. Unfortunately, this is also one of its weaknesses, leading to a large amount building up over time. The Python module python-env was written to try to deal with this issue. The conda packaging system also understands creating isolated Python environments. This way, you can have the best of both worlds: easier Python module management combined with easier Python environment management. You can get a list of all of the environments conda knows about with the following command:

```
conda env list
```

If you have just installed Berryconda, the only environment listed is the root environment. Now, let's say that you needed to start a new code base, developing the software for a big new project. You could create a new empty environment with the following command:

```
conda create --name deathstar
```

This command creates a new subdirectory within the envs subdirectory inside the Berryconda environment. If you rerun the environment list command, you will now see both the

root environment and the deathstar environment, with the latter tagged by an asterisk as the currently active one. You can now install only the modules you require for this specific software project by activating it and running the usual conda commands. For example, the following commands would install ipython, along with all of its dependencies, within the deathstar environment.

```
source activate deathstar
conda install ipython
```

You can leave a given environment by running the deactivate script to reset all of the environment variables. If you have already put together a basic environment that you want to use as a starting point for a new environment, you can clone it as your boilerplate. The clone option to the create command comes in handy for such a task.

```
conda create --name deathstar2 --clone deathstar
```

If the reason for your software project goes away, you can always clean up old environments with the remove command in conda.

```
conda --remove deathstar --all
```

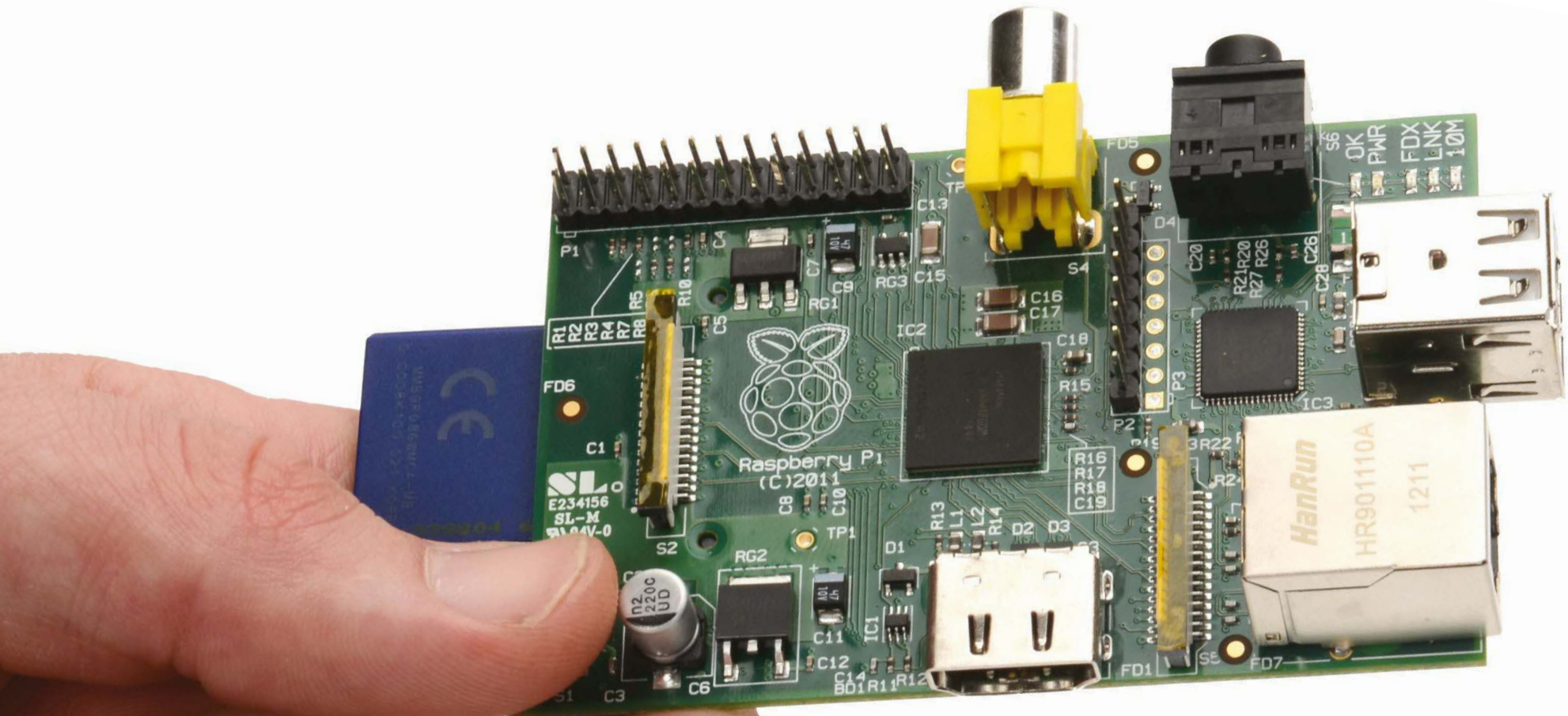
The command option --all ensures that all portions of the environment are deleted during the removal process.

Now, you should have all of the information you need to be able to work in isolated environments for your Raspberry Pi projects. This is especially useful in the development stages of a new project. You can get only the modules you need for a given project.

Talking Pi

Join the conversation at...

@linuxusermag | Linux User & Developer | @ RasPi@futurenet.com



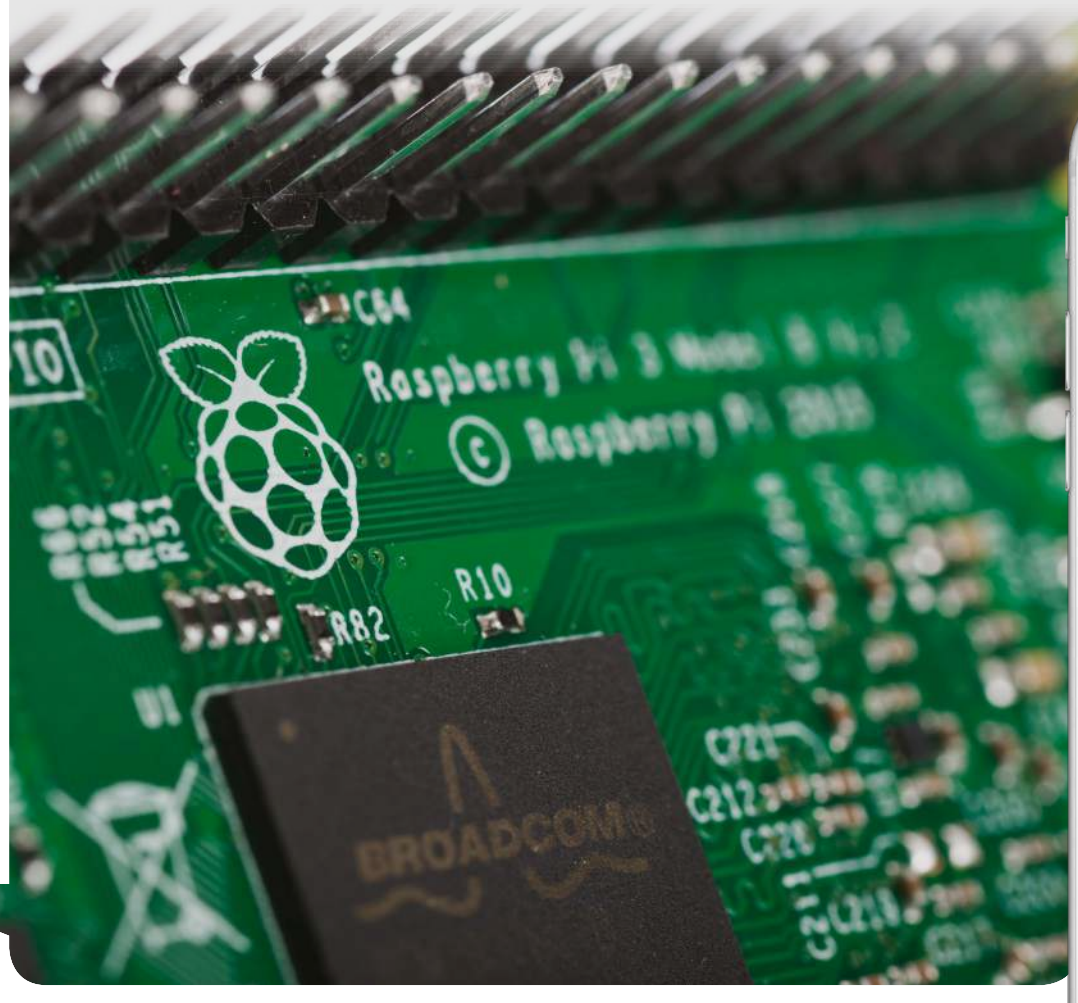
The Raspberry Pi encourages a lot of hands-on work and this means it doesn't quite work like – or as easily as – your laser focus-tested smartphone interface, or even a normal computer. So we're answering your burning Raspberry Pi questions each issue – get in touch with us on Twitter, Facebook or by email.

How can I display a message in the Minecraft environment?
Errol_slymm via email

You can display messages in the Minecraft environment to keep players up to date with what is happening. Use the code, mc.postToChat() and place the message to be displayed in-between the two brackets:

```
from mcpi import minecraft
mc = minecraft.Minecraft.create()
mc.postToChat("Hello world")
```

Messages are also useful for relaying data back to the player, for example, your current location or the block ID which you are stood on, or how far away from a particular block you are standing.



Keep up with the latest Raspberry Pi news by following @LinuxUserMag on Twitter. Search for the hashtag #RasPiMag

Is it possible to make the Pi solar-powered?
Frank, via email

Yes indeed, you can now get hold of an elegant little add-on board, called PiJuice, that lets you take your projects off-grid and away from mains power sources. PiJuice is compliant with the Raspberry Pi HAT (Hardware Attached on Top) specification and makes use of a slim, off-the-shelf mobile phone battery, and some intelligent charging and power circuitry, to make your Pi truly portable. There's also a version called PiJuice Solar that enables solar recharging and is even capable of taking inputs from other renewable energy sources. Check out the PiJuice Instructables page: **bit.ly/1e2CoGE.**

What's the biggest SD card the Raspberry Pi supports?
SteveQ via email

Officially, the biggest SD card (or microSDHC card if you have Raspberry Pi 1B+, 2B, 3B) that's been tested by the Raspberry Pi team is a 32GB card. As for choosing an SD card, the best way to select is not to go for the absolute cheapest one. Go with something from a recognised brand, as the Pi will be slower with a card of a lower class. There is a list of officially tested cards on the eLinux site, look for cards of a specific size and/or class: **http://elinux.org/RPi_SD_cards**

Next issue

Get inspired **Expert advice** **Easy-to-follow guides**

The Pi's the limit! Control a drone with Raspi

Get this issue's source code at:
www.linuxuser.co.uk/raspicode